



Software Requirements Specification (SRS)

For: *MediConnect – Hospital Management System*

Department of Computer Science and Engineering

University of Rajshahi

Course: Software Engineering (CSE-3112)

Date: 27 October, 2025

Table of Contents

1. Introduction
 - 1.1 Purpose
 - 1.2 Scope
 - 1.3 Definitions, Acronyms, and Abbreviations
 - 1.4 References
 - 1.5 Overview
2. Overall Description
 - 2.1 Product Perspective
 - 2.2 Product Features
 - 2.3 User Classes and Characteristics
 - 2.4 Operating Environment
 - 2.5 Design and Implementation Constraints
 - 2.6 User Documentation
 - 2.7 Assumptions and Dependencies
3. Specific Requirements
 - 3.1 Functional Requirements
 - 3.2 Non-Functional Requirements
 - 3.3 External Interface Requirements
 - 3.4 System Models (Use Case, ER, UML Diagrams)

- 4. Appendices
 - 4.1 Database Schema Overview
 - 4.2 Future Enhancements
-

1. Introduction

1.1 Purpose

The purpose of this document is to specify the functional and non-functional requirements of the **MediConnect Hospital Management System (HMS)**.

This SRS serves as a reference for developers, testers, and stakeholders to ensure the system meets its objectives: simplifying hospital discovery, appointment booking, and resource management through a unified digital platform.

1.2 Scope

MediConnect is a full-stack, web-based system designed for healthcare service management in Bangladesh.

It enables patients to book doctor appointments online, doctors to manage their schedules, and hospital administrators to monitor ICU beds and departmental resources.

Main Features include:

- Hospital search and filtering by location or department
- Doctor profile viewing and appointment booking
- Real-time ICU bed availability tracking
- Doctor schedule and off-day management
- Hospital and admin dashboards with analytics

The system uses **React**, **Node.js**, **Express**, **SQLite**, **Drizzle ORM**, and **Passport.js**, and is accessible from any modern browser.

1.3 Definitions, Acronyms, and Abbreviations

Term	Description
HMS	Hospital Management System
API	Application Programming Interface
ORM	Object-Relational Mapping
CRUD	Create, Read, Update, Delete
ICU	Intensive Care Unit
UI/UX	User Interface / User Experience
SRS	Software Requirements Specification
DBMS	Database Management System

1.4 References

- IEEE Std 830-1998: Recommended Practice for Software Requirements Specifications
 - React.js Documentation (v18.3.1)
 - Node.js Official Documentation (v18+)
 - Drizzle ORM & SQLite Documentation
 - Passport.js Authentication Library Docs
-

1.5 Overview

The remainder of this SRS describes:

- The system's overall architecture
- Functional and non-functional requirements
- System behavior, constraints, and assumptions

- Diagrams illustrating data flow and user interaction
-

2. Overall Description

2.1 Product Perspective

MediConnect is an **independent web application** that connects multiple user roles — patients, doctors, and hospital admins — within one ecosystem.

It follows a **three-tier architecture**:





- **Frontend:** React + TypeScript + Tailwind
- **Backend:** Express.js + Node.js + Drizzle ORM
- **Database:** SQLite

External Systems Integration:


- Optional email/SMS APIs for notifications (future enhancement)
-

2.2 Product Features

For Patients

-  Search hospitals and filter by division, district, and department
-  View detailed doctor profiles (specialization, fees, experience)
-  Book appointments with automatic serial generation
-  View real-time ICU bed availability

For Doctors

-  Dashboard with appointment overview

- 🕒 Schedule management (weekly availability)
- 🚫 Off-day management with reasons
- ✅ Mark appointments as completed or cancel with confirmation

For Hospital Administrators

- 🏥 Manage hospital info, departments, and doctors
- 🛏 Update ICU bed status in real time
- 📊 Monitor department-wise doctor distribution

2.3 User Classes and Characteristics

User Type	Description	Technical Level
Patient	General user searching and booking schedule info	Basic
Doctor	Medical professionals managing schedules	Intermediate
Hospital Admin	Manages hospital-level data	Intermediate
Developer	Maintains backend, database, and users	Advanced

2.4 Operating Environment

- **Frontend:** Browser (Chrome, Edge, Firefox)
- **Backend:** Node.js environment
- **Database:** SQLite (local) or PostgreSQL (production-ready)
- **OS Compatibility:** Ubuntu, Windows
- **Hosting (Deployment):** Vercel, Render, or VPS

2.5 Design and Implementation Constraints

- Authentication via Passport.js (session-based)
- SQLite database limits concurrent writes
- Must support responsive design for mobile devices
- Security compliance with the Bangladesh ICT Act 2006
- Open-source licensing (MIT License)

2.6 User Documentation

- Online documentation in README.md
- API endpoint documentation with examples
- Set up guide in `.env.example`
- Tutorial for the admin panel and user flows

2.7 Assumptions and Dependencies

- Users have stable internet access
- Hospitals and doctors provide accurate data
- Node.js runtime and dependencies installed correctly
- Future SMS/email APIs depend on third-party services

3. Specific Requirements

3.1 Functional Requirements

FR-1: User Authentication

- Users can register and log in based on their assigned roles (patient, doctor, or admin).
- Passwords must be hashed using encrypt.
- Session stored securely with `express-session`.

FR-2: Hospital Management

- Admin can add, edit, or delete hospitals.
- Each hospital must include name, address, contact, ICU info, and image URL.
- Real-time ICU bed count validation ($\text{available} \leq \text{total}$).

FR-3: Department Management

- Each hospital has multiple departments.
- Department must store `id`, `hospitalId`, `name`, and `description`.

FR-4: Doctor Management

- Doctors can set available days and time slots.
- Doctor details include `specialization`, `experience`, `consultationFee`, and `qualification`.
- Off-days can be marked with a reason.

FR-5: Appointment Management

- Patients can book appointments with selected doctors.
- Serial numbers are auto-generated sequentially.
- Prevent double-booking for the same time/date.

- Doctors can view and mark appointments as completed or canceled.

FR-6: ICU Bed Tracking

- Hospital admins can update ICU bed availability.
- The system enforces constraints to prevent exceeding total ICU beds.

FR-7: Search and Filter

- Patients can filter hospitals by **division**, **department**, or **ICU availability**.

FR-8: Role-Based Access Control

- Patients: Booking, view hospital info
- Doctors: Manage appointments and schedules
- Hospital Admin: Manage hospital data
- System Admin: Full CRUD access

3.2 Non-Functional Requirements

Category	Description
Performance	The system should respond within 2 seconds for common queries
Reliability	99% uptime for local deployment
Scalability	Support 100+ hospitals and 1,000+ users
Security	bcrypt password hashing, input validation, ORM-based queries
Usability	Intuitive UI, mobile-responsive
Maintainability	Modular codebase using TypeScript
Portability	Compatible with Ubuntu, Windows, and macOS

Backup & Recovery Manual and automated DB backup supported

Localization English (default), Bangla support planned

3.3 External Interface Requirements

3.3.1 User Interfaces

- **Login Page** – Authentication by username/password
- **Dashboard** – Role-based navigation
- **Appointment Page** – Calendar and serial generator
- **Hospital List Page** – Filter and sort options

3.3.2 Hardware Interfaces

- Any device capable of running a modern browser

3.3.3 Software Interfaces

- Node.js runtime, SQLite database, RESTful API endpoints

3.3.4 Communication Interfaces

- HTTPS protocol, JSON data exchange format
-

3.4 System Models

3.4.1 Use Case Diagram (Simplified)

Patient → [Search Hospitals] → [View Doctors] → [Choose schedule for Book Appointment]

Doctor → [Manage Schedule] → [View Appointments] → [Mark Complete]

Admin → [Manage Hospital Info] → [Manage ICU Beds] → [Monitor Stats]

3.4.2 Entity Relationship Diagram (ER Model)

Entities:

- Users (id, username, role, password)
- Hospitals (id, name, location, totalIcuBeds, availableIcuBeds)
- Departments (id, hospitalId, name, description)
- Doctors (id, userId, departmentId, specialization, fee)
- Appointments (id, doctorId, patientName, appointmentDate, serialNumber, status)
- Schedules (id, doctorId, dayOfWeek, startTime, endTime, maxPatients)
- DoctorHolidays (id, doctorId, date, reason)

(Relations: Hospital → Department → Doctor → Appointment)

4. Appendices

4.1 Database Schema Overview

Implemented with **Drizzle ORM + SQLite**

Each migration defines tables with validation, foreign keys, and seed data for hospitals, doctors, and users.

4.2 Future Enhancements

- Email and SMS notifications for appointments
- Payment gateway integration
- Telemedicine module
- Prescription and report management
- Doctor rating & review system

- Multi-language (English/Bangla) support
-

5. Conclusion

This Software Requirements Specification ensures that all project stakeholders share a unified understanding of **MediConnect's objectives, behavior, and constraints**.

It forms the foundation for design, implementation, and testing phases and adheres to the **IEEE 830 standard** for software documentation.

Final Note: MediConnect is designed to be a secure, extensible, and impactful healthcare solution tailored for the digital transformation of Bangladesh's hospital system.
